EXHIBIT D12

PROJECT PLAN

<<Insert Name of System here>>

Prepared by: <<Insert Name of Document Maker here>>

Release Date: 9-August-1996 <<Put your Document Release Date HERE>>

ABSTRACT: The project plan is the controlling document for the project. It is a living document, which serves to communicate scope, goals, objectives and resource requirements to (primarily) management, technical staff and the end-user organization. The project plan also serves to provide cost, schedule, activity and major milestone definitions, to monitor and track project progress and to outline the development approach. It should be a short, concise document that provides a broad overview of the project for management review and sign-off prior to undertaking strategy activities.

Organization 1

_____

Signatory-Title

Organization 2

_____

Signatory-Title

Bureau of Applications Development

_____
Signatory-Bureau Director

_____
Signatory-Systems Development Division Chief

_____
Signatory-Project Manager

_____
Signatory-Title of analyst who prepared document

<Insert>

Table of Contents

(Optional)

1 EXECUTIVE SUMMARY

This section should give a brief summary of the project for executives that includes a description of the project, the reason the project is being undertaken, who is involved with the project, and the scope of the project.

2 PROJECT OVERVIEW

The initial Project Plan is the first deliverable of any project. The Implementation Schedule Section is continually refined and updated throughout subsequent stages of the project and submitted with the deliverables of those stages as part of the Forward System Development Plan. The Project Manager performs this task.

2.1 PURPOSE

All project-related documents generated to date should be identified and reviewed to develop a project overview, which outlines the purpose and background of the project. The overview should describe the organization that the application is to serve, along with the business needs that are to be satisfied. It should be written from a "business benefit" perspective using the client's business terminology.

Three questions should be answered to ensure a thorough understanding of the project:

- Why are we doing the project? (New Legislation (include dates), Requested by User, etc.)
- What do we expect to accomplish? (List specific components of legislation that will be automated)
- What is not part of the project? (List any functions that may exist but that are not being automated at this time)

2.2 SCOPE

Definition of project scope and goals is one of the first activities in the project planning process. Project scope must be stated in terms that are unambiguous and understandable at both management and technical levels. Examples include specific business functions to be automated, the types and numbers of users and their location, required service levels, mandated deadlines and constraints, etc.

2.3 GOALS AND OBJECTIVES

Project goals establish overall direction for the project and describe how the end result will benefit the organization. As such, project goals are defined in terms of a solution to a business problem and are stated such that their

achievement is both realistic and measurable. Project goals must be in line with the general goals of the Department.

As an example, "The System shall provide a single source for client data, eliminating redundant data," is acceptable, where "Provide a robust system," is not acceptable.

2.4 CRITICAL SUCCESS FACTORS

The critical success factors of the project should also be identified to highlight the few key areas where "things must go right" in order for the project to be successful. Failure in any critical success factor area may affect attainment of project goals and ultimately jeopardize the success of the project.

Include other components or applications that may need to be in place prior to implementation, acquiring additional resources or hardware, user participation at all levels and site locations, timely review of documentation, or the receipt of required funding.

3 PROJECT ORGANIZATION

This section should include roles and responsibilities of all team members, how issues will be resolved and how project status will be reported. Also include a brief description any committee teams or organizations that may influence the project. Include the name of all individuals on the committee or organization and their designated role.

3.1 ROLES AND RESPONSIBILITIES

Roles and Responsibilities should be listed for all team members and project participants, following the format below.

Table 1: Roles and Responsibilities

Management	Their primary involvement is in making sure that the right people in the business give enough of their time to the project to make it a success, in resolving conflicts about what the project's goals should be, and in removing roadblocks to the project's timely completion.
Sponsoring User	Identify individuals who can provide details necessary to perform the tasks. Ensure the users are participating in a cooperative manner and are allocated the required time necessary to actively participate.
User	The users at this stage should be a selective sample of individuals who will serve as a true representation of the organization as a whole. These users must provide analysts with information necessary to clearly model the area of business and verify that the analyst interpretation is correct and true to form.
Project Leader	Responsible for planning and control, putting the plan into action, keeping all parties informed of plans, progress and issues, including establishing issue management and project status reporting procedures and managing the project team. It is important that the project leader maintain a short timeframe for this stage so as not to lose momentum. The key is to document the business and move on. The project leader must also keep in mind that the most cost effective and efficient solution may not include automated technology.
Analyst	Responsible for analyzing the business and identifying the terms of reference. Also, through interviews with the users, learn the business functions, procedures and tasks in detail enough to develop entity relationship diagrams, function hierarchy, system boundary definition and possible system architecture which clearly depict the area of business to be analyzed. Be sure to include all functions and entities, manual, as well as automated. While conducting interviews be sure to separate interviewee "needs" from "likes."

4 SYSTEM DEVELOPMENT APPROACH

The DEP system development methodology (SDM) encompasses proven technical methods, techniques, and tools to facilitate and support the development of high-quality software application systems:

Methods: The software development activities and deliverables comprising the DEP system development methodology and supporting standards, conventions, and guidelines.

Techniques: Data-driven modeling techniques to support analysis, design, development, and prototyping activities to establish and verify end-user requirements and develop systems based on the established requirements.

Tools: CASE (Computer-Aided Software Engineering) and software development tools to support and automate the methods and techniques comprising the software development process. These include Oracle CASE and software development tools, templates, and checklists.

4.1 SYSTEM DEVELOPMENT ACTIVITIES DELIVERABLES

The DEP SDM provides a framework for developing software applications by partitioning the development process into cohesive stages and activities in which proven techniques are applied to produce specific deliverables.

4.2 SYSTEM DEVELOPMENT ACTIVITIES AND DELIVERABLES

This section contains a list of activities to be performed during each of the stages of development, a list of the deliverables expected for each stage and a list of general criterion required for entry to and exit from each stage. The project manager may decide to customize the development approach at the project onset to reflect specifics of the project or the application to be built. This includes omission or consolidation of any development activities or deliverables. All deviations or customizations to the SDM (DEP Style Guide) must be identified by the Project Manager, approved by management and documented in the project plan.

Include in this section customizations that were made to the templates.

4.3 STRATEGY STAGE

The Strategy Stage contains tasks and activities designed to ensure an accurate portrayal of the client organization business needs and to guide the project team through analyzing the alternatives for an optimal solution. These activities will produce a Project Plan, a Strategy Report, issue management procedures and regularly scheduled status meetings. The primary activities comprising the strategy stage and the deliverables that are produced are outlined in the following table.

Table 2: Strategy Stage Activities and Deliverables

Activities

- Identify objectives, priorities, and critical success factors
- Document end-user needs, organizational direction, and constraints
- Define and quantify system scope
- Obtain management and user consensus and commitment
- Establish a common set of business terms
- Identify and analyze primary data entities
- Define primary business units, end-user classes/groups, and locations
- Develop a high-level entity-relationship model
- Develop a high-level function model
- Create a development plan for proceeding
- Review and obtain agreement with end-users
- Update/document the Forward Delivery Plan

Deliverables

- Strategy Report

4.3.1 ENTRY AND EXIT CRITERIA

Successful completion of each of the items in the following checklist indicates that the project is ready to move into the Analysis Stage of development.

Table 3: Quality Assurance Checklist: Strategy Stage

Criteria	**Measurement**
General	
Is the purpose of the system clearly defined?	YES[ ] NO[ ]
Are objectives stated in end-user "business terms"? Are they concise, measurable, and achievable?	YES[ ] NO[ ]
Are assumptions and constraints identified?	YES[ ] NO[ ]
Is the project approach clearly expressed so that all parties involved understand the next steps and what is expected of them?	YES[ ] NO[ ]
Is it clear how the new system will fit into the organization and benefit end-users?	YES[ ] NO[ ]
If the system is to replace an existing automated system, has the current system been evaluated and strengths and limitations identified?	YES[ ] NO[ ]
Are the critical success factors agreed and indeed critical?	YES[ ] NO[ ]
Were identified priorities and objectives agreed to at an Executive Management level?	YES[ ] NO[ ]
Has a sponsor been identified from the user organization?	YES[ ] NO[ ]
Are roles and responsibilities clearly defined for key project players?	YES[ ] NO[ ]
Has the Strategy Study been reviewed and agreed-to by end-user personnel?	YES[ ] NO[ ]
Have all entities, relationships, attributes, functions, and business units been completely and correctly entered into CASE*Dictionary?	YES[ ] NO[ ]
System Scope	
Has a technical review been performed by the Project Team on the entity relationship diagram, function hierarchy, and the Strategy Study?	YES[ ] NO[ ]
Is the system scope explicit and unambiguous?	YES[ ] NO[ ]
Is the scope described in terms of specific functions and data to be included in the system?	YES[ ] NO[ ]
Are all interfaces with other systems and databases, both internal and external to DEP, identified?	YES[ ] NO[ ]

Table 3: Quality Assurance Checklist: Strategy Stage *(continued)*

Criteria	Measurement
Is it obvious what is not included as part of the system scope?	YES[ ] NO[ ]
Entities	
Does each entity have a meaningful name that is unique within the application? Is the name a singular noun?	YES[ ] NO[ ]
Does the entity have a short name that is unique within the application?	YES[ ] NO[ ]
Does each entity have a concise, meaningful description?	YES[ ] NO[ ]
Does each entity have a unique identifier?	YES[ ] NO[ ]
Does each entity have at least two significant attributes?	YES[ ] NO[ ]
Do the sets of data specified by sub-type entities make up the total set of data represented by the supertype entity?	YES[ ] NO[ ]
Are any synonyms and homonyms identified for each entity?	YES[ ] NO[ ]
Are "global"/shared entities identified as such?	YES[ ] NO[ ]
Is each entity internal to the system or business area being studied?	YES[ ] NO[ ]
Does each entity have a plural name that is grammatically correct and matches the entity name?	YES[ ] NO[ ]
Does each entity have a concise shortname that accurately reflects the entity?	YES[ ] NO[ ]
Do all entity names conform with established naming conventions and standards?	YES[ ] NO[ ]
Have all entities been mapped to business units?	YES[ ] NO[ ]
Attributes	
Does each attribute have a name that reflects its purpose, and is unique within that entity?	YES[ ] NO[ ]
Does each attribute have a concise and meaningful description?	YES[ ] NO[ ]
Is the optional-ity (required vs. not required) of each attribute specified?	YES[ ] NO[ ]
If any attribute forms part of the unique identifier of an entity is this recorded?	YES[ ] NO[ ]
Relationships	
Is each end of the relationship named?	YES[ ] NO[ ]
Is the optional-ity and cardinality of each end specified?	YES[ ] NO[ ]

If any relationship forms part of the unique identifier of an entity, is this recorded?	YES[ ] NO[ ]
Does each relationship follow the naming syntax outlined in the Barker book?	YES[ ] NO[ ]
Does the relationship make sense when read aloud?	YES[ ] NO[ ]

Functions

Does each function have a concise and meaningful description that begins with a verb?	YES[ ] NO[ ]
Is the sum of the functionality of the child functions equal to the functionality defined for the parent node?	YES[ ] NO[ ]
Are there any parent nodes with excessive numbers of children (i.e. 10 or more) that could be further split?	YES[ ] NO[ ]
Do any functions refer to objects that are not entities, synonyms or entities, attributes, etc.?	YES[ ] NO[ ]
Have all elementary functions been identified as such?	YES[ ] NO[ ]
Have all functions been mapped to business units?	YES[ ] NO[ ]

Business Units

Have the primary user organizations and locations been identified (i.e. Region, Bureau, Program, etc.?)	YES[ ] NO[ ]
Have the prospective users groups or classes been identified?	YES[ ] NO[ ]
Have equipment needs for the respective users/location been identified (i.e. network connections/terminals, printers, etc.)	YES[ ] NO[ ]
Have all business units been mapped to entities and functions?	YES[ ] NO[ ]

4.4 ANALYSIS STAGE

Findings from the Strategy Stage are verified and expanded upon during this stage. Business accuracy, feasibility and a sound foundation for design are ensured within the scope of the organization. The Requirements Specification establishes the foundation for all subsequent design and development work. The following table outlines the primary activities and deliverables comprising the analysis stage.

Table 4: Analysis Stage Activities and Deliverables

Activities
Develop a fully-attributed entity-relationship model with detailed entity, attribute, and domain definitions

Table 4: Analysis Stage Activities and Deliverables *(continued)*

Activities (continued)
Develop a fully-decomposed function hierarchy with detailed function definitions

Cross-check the data and function models using the Oracle CASE matrix diagrammers

Identify preliminary data volumes, function/transaction frequencies, performance requirements, and estimated database sizing

Identify user types and access/security requirements

Define data conversion requirements

Identify operational requirements such as backup, recovery auditing and control, etc.

Define user acceptance criteria

Identify initial installation and production transition needs such as training, data cut-over, supplies, etc.

Identify design assumptions and constraints

Review and obtain agreement with end-users

Update/document the Forward Delivery Plan

Deliverables
Requirements Specification (includes Data Conversion Plan)

4.5 DESIGN STAGE

The Design Stage focuses on "how" the system will be implemented to achieve the requirements specified in the Requirements Specification. A Database Design is created from the Entity Relationship Model. Functions will be translated into modules and manual procedures, with required audit/control features and back-up/recovery points. Screens, reports and module linkages will be derived. Function usage will be employed to drive communications architecture design. Prototyping may be used to help make decisions on areas of doubt, but must be seen as a technique and not an end in itself. Program Specifications and a System Test Plan are also produced. The information gained in this stage is used to confirm the transition strategy.

This section, ESPECIALLY, must address the CASE tool and the requirements of the generator and what outputs will fill the requirements - also how prototyping fits in. The primary activities and deliverables comprising the Design Stage are outlined in the following table.

Table 5: Design Stage Activities and Deliverables

Activities

- Define system architecture
- Produce logical and physical database design
- Develop detailed module/program specifications
- Define detailed database sizing and storage parameters
- Design data conversion programs, utilities, and data Clean-up procedures
- Define required manual processes and procedures
- Develop end-user training plan
- Develop "first-cut" users guide
- Develop system test plan
- Develop transition strategy for installation, acceptance, training, data take-on, and cut-over
- Update/document the Forward Delivery Plan

Deliverables

- Design Report
- System Test Plan
- Transition/Training Plan

4.6 BUILD STAGE

During the Build Stage programs are coded and tested using the appropriate tools. The purpose of this stage is to develop and test individual modules as specified in the Design Stage, test individual application modules, perform system testing, revise the system transition plan and finalize the user training plan.

The User Documentation Stage will deliver the user manuals and operations hand-over documentation. These should include what is expected of the user and the meaning of, and appropriate reaction to, each error produced. A tutorial would be an efficient way in which users can gain an understanding of how the system works and how it serves the organization. The User's Guide should be written using VAX Document or ALL-IN-1 Word Processing (in a shared file folder format), unless it is written and maintained by the user, in which case the user can choose the word processing packaged to be utilized.

The primary activities comprising the Build and User Documentation Stages are outlined in the following table.

Table 6: Build and User Documentation Stage Activities and Deliverables

Activities

- Deliver and install operational hardware and software

- Develop system programs/modules
- Perform module, integration, and system-level testing
- Develop and implement manual procedures and paper forms (if applicable)
- Complete end-user and operations documentation
- Construct production database
- Refine transition plan

Deliverables

- Completed & tested modules
- Detailed Module Specifications
- Fully tested system
- End-user and operations documentation

4.7 TRANSITION STAGE

All tasks necessary for implementation are performed during this stage. This stage focuses on data conversion and take-on, system integration and testing, implementing data archiving and backup procedures, establishing a support plan and/or help desk, providing user training and obtaining user acceptance. An initial period of support for the system is provided during this stage. Transition must be accomplished with minimum disruption to the business.

A draft transition plan is developed during the analysis stage and is refined and finalized during the design and build stages. The Project Manager and Client Management jointly prepare it. For especially large projects, initial development of this plan may start in the Strategy Stage.

The transition plan should outline the tasks required to support the delivery and production operation of the new system in the client organization. The following list details the Transition Strategy components outlined in CASE*Method based on specifics of DEP's development environment:

- System acceptance criteria, acceptance testing and system sign-off.
- Management and coordination of effort to support/maintain dual systems during the transition to the new application platform.
- Backup/archiving procedures and other operational support issues.

Outlined in the table below are the primary activities and deliverables of the Transition Stage.

Table 7: Transition Stage Activities and Deliverables

Activities

- Perform data conversion and data clean-up
- Install production database and perform production data loading
- Deliver end-user and operations documentation

- Train users, operations staff, and technical support staff
- Perform acceptance testing
- Implement support help desk and support facilities
- Implement change request and problem reporting procedures
- Identify and order supplies (consumables)

Deliverables

- Installed, operational, and accepted system
- Completed and approved documentation
- Trained users and support staff
- Help desk and end-user support
- Operations procedures
- Acceptance test results
- Change request and problem reporting procedures

4.8 PRODUCTION STAGE

The production stage ensures smooth running of the system with minimum intervention from operations or support staff. During this stage, the systems usage and performance are monitored at each live running site. This stage also provides for evaluating and implementing change request, performing data backup, archival and recovery and conducting spot review and integrity checks.

Table 8: Production Stage Activities and Deliverables

Activities

- Produce backup and archive files/tapes
- Establish change control and problem reporting logs
- Evaluate and implement change requests
- Monitor and tune application software and database
- Train new users
- Provide access to new users

Deliverables

- Change requests/problem reports
- Performance reports
- On-going training and support

5 CASE AND SOFTWARE DEVELOPEMNT TOOLS

The system development approach is predicated on the use of automated software tools, documentation templates, development checklists, and procedures to automate, manage, and support the software development process. Integration of software tools into the development process facilitates development of high-quality software systems in a timely and cost-effective manner. The following standard tools are to be utilized to support all aspects of a DEP project.

Microsoft Project. The project planning and management tool to support task definition, scheduling, resource loading, cost monitoring, and project status assessment and reporting activities.

Oracle CASE*Designer. The front-end analysis tool for developing graphical models of the system being developed. This tool provides the capability to develop detailed data and function models through an easy-to-use graphical user interface (GUI). CASE*Designer is integrated with CASE*Dictionary, thus, providing the capability to populate dictionary with object definitions created using CASE*Designer. CASE*Designer is used during the strategy and analysis stages of the system development methodology.

Oracle CASE*Dictionary. The central repository for all analysis, detailed design, and implementation-related information pertaining to each system. In addition to providing crosschecking and reporting facilities, CASE*Dictionary provides utilities to expedite the application and database design process. CASE*Dictionary supports all stages of the SDM, from strategy through production.

Oracle CASE*Generator(s). CASE*Generator is used to automate the build stage (coding stage) of the development process to the extent possible. The generator product produces interactive screens, reports, and menus based on the require-ments and detailed design information in CASE*Dictionary. CASE*Generator supports the system prototyping process by generating screens and reports early on in the development process.

Oracle SQL*Forms. The general-purpose tool for developing and executing interactive forms-based applications (screens). SQL*Forms is a 4GL application development tool that provides an overall application structure and framework for creating, displaying, and updating data residing in an Oracle database. SQL*Forms screens are the primary user interface to data residing in the Oracle database.

Oracle SQL*ReportWriter. SQL*ReportWriter is a database reporting tool that enables application developers to create fully formatted, multi-part reports. SQL*ReportWriter is the primary tool for developing standard formatted reports.

Oracle SQL*Plus. SQL*Plus is a database reporting tool for producing formatted reports and writing command procedures to manage information on an ORACLE database.)

Oracle SQL*Menu. SQL*Menu is a development tool for creating menus that provide access to other Oracle products such as SQL*Forms screens and SQL*ReportWriter reports.

Oracle SQL*Loader. SQL*Loader is a data tool for moving data in external files into tables in an ORACLE database. It loads data in a variety of forms, performs filtering (selectively loading records based upon the data values), and can load multiple tables simultaneously.

Oracle PL/SQL. PL/SQL is an application development tool which allows the use of procedural techniques, such as looping and branching to process data. PL/SQL combines the data manipulating power of SQL with the data processing power of procedural languages.

VAX DEC Test Manager (DTM). VAX DEC/Test Manager organizes software tests and automates the method used to run tests and evaluate test results. DTM can be used to "record" interactive dialogues between an end-user and a SQL*Forms application for comparison against subsequent dialogues after a change has been made to the form. Likewise, DTM can be used to test SQL*ReportWriter results when changes are made to a report.

VAX DEC/Code Management System (CMS). CMS is an automated software library system that provides an efficient method for storing and maintaining all project documents and source files. It provides centralized control over all project deliverables by limiting access and tracking all changes to library contents.

VAX DEC/Module Management System (MMS). MMS automates and simplifies the building of large and complex software systems. It efficiently constructs the "executable" components of a software application from source input files. MMS optimizes the build process by rebuilding only those components that have changes since the system was last built.

SDM Document Templates. Document templates are created for each of the SDM deliverable documents to be produced during the course of each system development project. These templates are electronic files containing a standard document format, detailed descriptions of what is to be placed in each respective section, and quality control checklists to help ensure that all tasks are complete and concise. Document templates expedite the document "writing" process by providing a standard document format and supporting boilerplate text.

System Development Style Guide. The project style guide is a "desk-top reference" provided to each project developer. The style guide outlines details of the system development methodology (SDM) and specifies project standards, conventions, and guidelines to be adhered to throughout each stage of the development process.

5.1 SYSTEM PROTOTYPING

Prototyping will be used as part of this development effort. Prototyping is used to elicit and verify requirements with end users. There are data requirements that are collected and verified by the display of the fields on various screens.

Functional requirements are developed and verified by analyzing the work performed on and by the various screens. Active user involvement throughout the prototyping process is critical to the development and delivery of application systems that satisfy user requirements.

Prototyping allows users direct involvement in the work flow of the system. This includes movements from menu to menu, menu to screen, screen to screen and even the flow or movement within a single screen as an operator moves from field to field. The prototyping effort is also used to help develop training materials. The users can provide guidance as to which aspects of a new system will require special attention when full scale training occurs, as well as insight to how users may relate the new system to any old systems or ways of doing business.

6 MANAGEMENT PROCESSES

The complexity and dynamic nature of system engineering and implementation processes makes effective project management mandatory. The Department's Style Guide will be used as a framework for the software development process. This development framework will be supplemented with a set of proven project management methods and processes to monitor, manage and control all aspects of the project. In addition, the project team will rely on a broad set of automated support tools to assist in this effort and facilitate achievement of the project objectives.

This section outlines each of the management processes that provide the most effective pay-back in terms of schedule, cost and DEP satisfaction. These processes include:

- Status Reporting
- Issue Resolution and Problem Reporting
- Change Management
- Quality Assurance
- Configuration Management
- Project Monitoring, Control and Reporting
- Application Acceptance and Project Closure
- Time Reporting and Tracking

6.1 STATUS REPORTING

The Project Status Report is the formal vehicle for communicating progress against the plan and schedule, and to provide status information pertaining to outstanding issues, problems, and the overall work effort expended for the reporting period. It is prepared on a weekly or biweekly basis by the Project with input from project team members assigned to project tasks. The frequency with which status reports are submitted and the distribution list to whom the report is sent is established by the Project Manager and documented here. All Project Status Reports are maintained in the Project Notebook and should adhere to outline below:

a. Sample Status Report Outline

1. Activities and Accomplishments

2. Problems, Issues or Concerns

3. Plans for Next Period

6.2 ISSUE MANAGEMENT

An issue is a circumstance that prevents (or limits the effectiveness) of a team member or an end-user from performing their job on time or within established quality standards. Issues include technical problems, awaiting answers for unanswered questions regarding desired functionality, and people/organizational problems.

The process for identifying, documenting, and resolving issues is established at the beginning of the project and must be communicated to project team members, management, and the client organization. The issue resolution/escalation process should be documented here.

Issues that do arise during the course of the project should be communicated to management in the Project Status Reports and documented in the Project Notebook. Issues must be closely monitored to ensure that appropriate action is taken to bring them to closure.

A standard form should be developed to document issues and issue status-related information in a consistent manner. Information tracked should include:

- Name of person who identified the issue and date recorded
- Description of the issue
- The impact (or potential impact) of the issue
- The corrective course of action
- Disposition of the Issue/Resolution date

6.3 CHANGE CONTROL PROCEDURES

Change control procedures are established during the Strategy Stage and are enforced throughout the remainder of the project by the Project Manager.

Change Control refers to the process for managing change throughout the software life cycle. It is a software quality assurance activity applied throughout the development and production use of a software application. Change control is a component of a larger Software Quality Assurance activity; Configuration Management. It defines the process for managing requested changes to project scope, deliverables, or milestones that would affect the project cost, schedule, quality, or conformance of the deliverables to agreed specifications.

A standard Change Request Form should be used to process all change requests submitted by the client organization. This form must be completed by the end-user organization for each requested change and submitted to the Project Manager. The impact of the requested change to overall project cost and schedule is assessed by the Project Manager with input from the project team. If there is significant impact to cost, schedule, or client satisfaction, the

Project Manager must obtain approval and sign-off from both his management and end-user management prior to implementing the request. All change request forms should be maintained in the Project Notebook. Change control procedures are established by the Project Manager at the beginning of the project and are documented in the Project Plan.

The change control process involves the following:

- A change request form used to document the requested change with a description of the change, the reason for the change, etc..
- An organizational body for formally evaluating, approving or disapproving proposed changes, and prioritizing approved changes for implementation.
- Procedures for incorporating and properly documenting changes and appraising appropriate personnel of the changes.
- Configuration Reviews and Audits.

6.4 QUALITY ASSURANCE

Quality assurance (QA) it is critical to the success of the project. QA is an "umbrella activity" comprised of quality controls and checkpoints applied throughout the software development process to ensure that project deliverables and the process used to develop them satisfy established quality criteria and standards. Broadly stated, the goals of QA are:

- To improve software quality by appropriately monitoring and assessing both the software products and the development and management processes used to produce it.
- To ensure full compliance with established standards and procedures for the software products and the development and management processes.
- To ensure that any inadequacies in the software products, the processes, or the standards are identified, brought to management's attention, and resolved.

Quality assurance takes place throughout the entire life of the project, integrating the project management process and the software development process. The quality process begins during project start-up where quality standards, methods, and procedures are defined and implemented. These QA standards and methods are incorporated into the software development methodology. During each stage of the SDM, the required level of quality is designed and built into the respective application system.

A continuous assessment of quality as the application system is developed ensures that corrections can be made immediately, at every step in the development process.

The following sections provide an overview to the quality assurance and quality management practices to be employed throughout the life of the project.

6.5 QUALITY DEFINITION

Quality is defined as conformance to explicitly stated functional and performance requirements, explicitly documented development standards, and implicit characteristics that are expected of all professionally developed software such as correctness, maintainability, and reliability. Quality, required of each development project, is defined in documents such as the requirements specification and in management processes such as change control and configuration management. During the quality definition process, quality is specified by means of well-defined attributes, target values, and quality metrics. Examples of quality attributes include functionality, performance, reliability, maintainability, flexibility, and efficiency. The quality plan references specific quality requirements and ensures that they are defined, measurable, and verifiable.

6.6 QUALITY MANAGEMENT

Quality management involves definition, execution, and maintenance of the quality plan to achieve DEP requirements. The quality plan defines quality standards, activities, and resources necessary to ensure that quality is built into the software products, the activities used to develop the software, and the processes used to manage the project. The Quality Manager is responsible for ensuring that quality standards are defined, documented, and achieved. Quality management includes:

Software_Product_Quality - Ensuring that all project deliverables satisfy both implicit and explicit DEP standards.

Development_Process_Quality - Establishing checkpoints and controls to ensure that all deliverables are developed in accordance with the DEP System Development Methodology (SDM).

Management_Quality - Implementing project monitoring and control processes and procedures, managing change, and resolving issues and problems.

6.7 QUALITY PLAN

The quality plan is prepared as a first step in the quality assurance process. The purpose of the quality plan is to establish, document, and communicate the quality standards, procedures, and activities to be applied throughout each stage of the software development process. Management of the Quality Plan involves the following:

- Development of Quality Plan. Preparation of the quality plan begins during the initial stages of the project. The quality plan defines the quality requirements and standards for both the software systems to be produced and for the process used to develop them. In addition, it identifies activities necessary to meet established requirements and standards and outlines quality assurance responsibilities and deliverables.

- Execution of Quality Plan. All members of the project team are responsible for performing the quality activities defined in the quality plan. The quality manager is responsible for ensuring that the quality plan is carried out in order to meet established quality requirements.

- Maintenance of Quality Plan. The quality plan is the central reference document for all quality assurance activities. It is a living document that will be updated as required to reflect changes in the project.

  Specifically, the quality plan addresses:

  - Quality attributes, factors, and metrics to specify and measure quality characteristics of the software.

  - Quality activities such as structured walkthroughs, formal technical reviews, audits, inspections, and checklists to ensure that all project deliverables satisfy established quality standards and requirements.

  - Quality procedures such as issue and problem reporting, change control, and software configuration management.

  - Quality assurance organization, roles, and responsibilities.

  - The Quality Assurance Manager has overall responsibility for developing, executing, and maintaining the quality plan.

6.8 QUALITY ASSURANCE ACTIVITIES AND SCHEDULE

Quality will be measured and managed primarily by means of product walkthroughs and reviews. Each deliverable will be reviewed against the standards documented in the quality plan; reviews will further ensure that all functional and technical requirements have been fully satisfied.

In addition to quality reviews, a comprehensive, multi-tiered testing strategy should be applied to ensure quality and demonstrate application performance. Each of these activities are described in the following sections.

6.8.1CONDUCT REVIEWS

The quality assurance process is comprised of key review functions applied throughout each stage of the system development process. Continuous assessment of quality as it is developed ensures that corrections can be made quickly through the issue management process, at every step in the development process.

6.8.1.1 Structured Walkthroughs

Structured walkthroughs are a group review within the project team of any work product such as a data model, module design, module code, test case, algorithm, database design, or any other intermediate work product that will become part of a formal deliverable. The purpose of the walkthrough is to identify any defects or potential problem areas as early in the development process as possible. The earlier a defect is identified and corrected, the less impact there is to both cost and schedule. Structured walkthroughs enable the project team to take necessary and timely corrective action.

The results of the walkthrough are documented and forwarded to walkthrough participants. If follow-up action is required, the results are forwarded to the respective project team leader for resolution. Participation in structured walkthroughs is vital to ensure that each system satisfies user needs.

6.8.1.2 Technical Reviews

After all of the key elements of a deliverable are first reviewed by means of walkthroughs, the deliverable as a whole is scrutinized during formal technical review. The purpose of the technical review is to evaluate a deliverable (or set of deliverables) and provide management with evidence that the development of the deliverable(s) is being performed according to established standards, guidelines, and specifications. The fundamental reason for conducting these reviews is to establish a level of confidence in the technical integrity of the solution, not to meet a milestone. Technical reviews are triggered by completion of a document deliverable such as a requirements specification or successful completion of system testing. Participants of the technical review include representatives from the user organization, BIS management and technical staff, and when required, auditors from the Comptroller's office, as well as members from the project team. As such, technical reviews are more formal in nature, with a predefined agenda, and are planned longer in advance than walkthroughs. Review materials are submitted to participants at least five days in advance of the review date to allow adequate time for evaluation and feedback. Deficiencies and deviations identified during the review are recorded and forwarded to the respective team leader for resolution.

6.8.1.3 Phase Reviews

The project deliverables for each respective development phase are ready for management review and approval only after being subject to peer review walkthroughs and technical reviews. At this level, the deliverable receives a final approval decision before proceeding with the subsequent stage of the development process. The final deliverable documentation is presented to the BIS Bureau Director, the Development Division Chief and designated User Management for review and approval. The management review team evaluates the deliverable package to ensure that it is consistent with the project schedule, authorizes funds, authorizes schedule changes as required, verifies user commitment and participation, and resolves any outstanding issues before proceeding with sign-off. DEP management sign-off during the phase review is a milestone in the development process and indicates formal transition to the next stage of the development process.

6.8.1.4 Audits

Audits are an integral part of the quality process to ensure the success of the development effort. The audit review is intended to provide an independent analytical check on the software products and the processes used to create them. In addition, the audit review makes constructive recommendations for handling identified issues and problems.

The QA manager will conduct periodic audits of individual projects. The purpose of these audits is to:

- Ensure that policies, procedures, and standards established in the quality plan are adhered to by each project team.
- Develop and implement corrective actions should they be required.

6.8.2IMPLEMENT TESTING STRATEGY

Testing is a "safety net" that ensures that each system will perform in accordance with the requirements documented in the requirements specification. A testing strategy utilizes a structured testing methodology to identify defects and problem areas, maximize developer productivity, expedite the testing process, and above all, ensure delivery of an application that fully satisfies the Department's requirements.

The testing process involves test plan development, test case (data) design and creation, test script design and creation, test execution, and test results evaluation. The testing process begins during the design stage and will involve four levels of testing:

- **Unit Testing.** Verifies module-level compliance with established functional requirements.
- **Integration Testing.** Verifies inter-module interfaces, intersystem interfaces, and interfaces with other DER databases.
- **System Testing.** Verifies that the system as a whole satisfies input/output specifications and ensures error-free functional and performance requirements.
- **Acceptance Testing.** Verifies full compliance with functional requirements and acceptance criteria as stated in the requirements specification.

6.8.3IDENTIFY AND MANAGE RISK

Risk management is a critical aspect of the quality assurance process. As part of the QA function, risk factors will be identified and brought to the attention of the Project Director and the appropriate project team. Specific recommendations to prevent or minimize risk will then be made. Risk identification is part of the on-going risk management process described in the "Project Dependencies, Risks, and Contingencies" section of this document.

6.9 ESTABLISH QUALITY METRICS

The QA function will define quality attributes, factors, and metrics to specify and measure quality characteristics of the software in each aspect of the development life cycle including analysis, design, development, testing, and implementation.

6.9.1 PROVIDE RECOMMENDATIONS FOR IMPROVEMENT

The Quality Manager will be available for consultation and guidance on any and all project management and development process-related issues and concerns. Based on scheduled and ad-hoc re-views, specific recommendations will be made for improvements in product and process quality.

6.9.2 REPORT PROGRESS TO PROJECT MANAGEMENT

Recommendations arising from the quality assurance process will be provided to the Project Manager and Team Leader(s) on a regular basis in the form of written feedback and reports.

6.9.3QUALITY ASSURANCE ACTIVITIES SCHEDULE

A schedule of the major quality activities should be provided here.

6.10 ROLES AND RESPONSIBILITIES

The Quality Assurance Manager is responsible for defining the quality requirements, planning the quality activities, and performing many of these activities throughout the project.

As a key point of contact for all quality-related matters, the QA Manager is responsible for developing and executing the Quality Plan and working with each development team to ensure that established quality standards, guidelines, and procedures are used and adhered to throughout all phases of the project in addition, the QA Manager monitors the development process and the resulting deliverables to identify trends that may indicate a need for new standards and procedures or revisions to existing ones.

Quality, however, is not the responsibility of the QA Manager alone. Rather, quality is the responsibility of each and every individual on the project team. To ensure quality results each team member will adhere to the system development methodology (SDM) and supporting standards and procedures described in the Style Guide. In addition, each project team is responsible for:

Ensuring that quality processes and procedures are incorporated into day-to-day development activities.

Implementing, reviewing, verifying, and validating that each deliverable adheres to established standards, conventions, and procedures.
Taking corrective action to fix any and all deficiencies and/or problems that are identified during the review process.

6.11 QUALITY ASSURANCE TOOLS

The following tools will be provided to each project team member for performing quality-related activities;

QA Checklists. The quality assessment process will be enhanced through the use of quality control checklists established for each stage of the system development process. Checklists help ensure that tasks are completed in a structured manner and enable the work to be reviewed prior to walkthroughs and technical reviews.

SDM Document Templates. Document templates are created for each of the SDM deliverable documents to be produced during the course of each system development project. These templates are electronic files containing a standard document format, detailed descriptions of what is to be placed in each respective section, and quality control checklists to help ensure that all tasks are complete and concise.

Project Style Guide. The project style guide is a "desk-top reference" provided to each project developer. The style guide outlines details of the system development methodology (SDM) and specifies quality standards, conventions, and guidelines to be adhered to throughout each stage of the development process.

Requirements Trace-ability Matrix. A trace-ability matrix is created and included as part of the acceptance test plan. The trace-ability matrix maps specific requirements identified in requirements specification to individual tests in the acceptance test package to verify expected functionality.

6.12 PROJECT DEPENDENCIES, RISKS AND CONTINGENCIES

Risk analysis is an iterative process that is performed throughout a project's life cycle. Risk analysis examines the risk and its potential impact to the project and defines actions to eliminate or to mitigate the impact of that risk should it materialize.

Risk exists in all business transactions. During the course of the project there is the chance that something will go wrong even under tight management and control, and the project team will have to resolve the problems that result. Our approach to risk management includes identifying risk early in the project life cycle, assessing and prioritizing risk, planning for risk reduction and contingency activities, and monitoring risk throughout the project life cycle.

Risk can originate from a wide variety of sources, including:

Schedule or budget constraints
Requirements specifications
Requirements stability/instability
Contractor performance
Subcontractor performance
Customer performance
Technology requirements
Performance requirements
Integration requirements
Resource availability

Risks are associated with all projects. Some of these risks are "generic," in that they are inherent to the process of providing the integrated systems solution. We have developed a formal risk management process that we will use to overcome many generic risks associated with the project. This risk process will integrate sound business policies and procedures. Together, these constitute a repeatable process for ensuring the successful delivery of a quality solution to the Commonwealth.

The following table delineates the techniques and steps that will be used as part of our risk management approach. Note that there is no specific beginning or end to the risk management process. It is continuous throughout the life of the project.

Table 9: Risk Management Techniques

Major Process Step	**Technique**	**Detailed Actions**
Identifying Risk	Pinpointing Risk	All project members: Evaluate technical risk Evaluate third party risk Evaluate support and training risk Evaluate business risk Evaluate other risk
	Highlighting issues	All project team members: Identify and document areas that are putting the project at risk

Table 9: Risk Management Techniques *(continued)*

Major Process Step	**Technique**	**Detailed Actions**
Identifying Risk *(continued)*	Highlighting Issues *(continued)*	Identify areas requiring additional information
		information to the team to raise visibility
		Provide information to the team to raise visibility
		Use internal management and the Project Management Team as a vehicle for issue awareness and resolution
Assessing and prioritizing risks	Assess likelihood of a risk event occurring	Project Managers and Sponsors:
		Evaluate the potential of risk occurring
		Classify risk as LOW, MEDIUM, or HIGH probability of occurrence
	Assess potential severity	Project Managers and Sponsors determine:
		What is the cost to recover?
		What is the impact to the customer?
	Prioritizing risks to be managed	Project Managers:
		Map risks by probability and severity
		Determine the urgency (if risk occurs, how soon will it impact?)
		Prioritize by severity, likelihood, and urgency
Planning risk reduction and contingency activities		All Project members strive to:
	Planning risk reduction activities	Reduce uncertainties about given risks
		Reduce the consequences of a risk

Table 9: Risk Management Techniques *(continued)*

		Avoid the situation causing the risk Transfer the risk (with consent)
	Planning risk contingency activities	By using extra hardware, software, and services Identifying and qualifying new third parties where appropriate Rework, including redesign where required Increase Resources Make more efficient use of existing resources
Monitoring risk	Monitoring	All project members: Identify and document any changes in risk Assess the effectiveness of risk reduction activities risk reduction activities Review risk contingency plans for their current effectiveness

6.13 DEPENDENCIES ON OUTSIDE PROJECTS/EVENTS

Dependencies to other projects/events should be placed here. Included may be some of the items below.

6.14 DEPENDENCIES ON THIS PROJECT BY OUTSIDE PROJECTS/EVENTS

Dependencies on a project by external development groups should be included here. i.e., Schedule Dependencies, (Data Sharing & CASE)

6.14.1. RISKS AND CONTINGENCIES

Each risk should be stated in the following terms:

A brief description of the event associated with the risk.
The probability of the risk event occurring (% or High, Medium, Low).

The impact to the project if the event occurs (~hours lost).
The preventive measures that could minimize or eliminate the risk.

Contingency measures that can be taken if the event associated with the risk occurs.

The estimated number of project hours lost, as a result of risk, can be calculated by multiplying the estimated number of hours lost if the event occurs (in hours) by the estimated probability of the event occurring (expressed as a percent), then summing these risk hours for all project risk areas. Risk hours can be factored into the estimates of individual project activities having high risk potential as a proactive contingency measure.

The greatest or most prominent risks must be brought to Management's attention and documented in the Project Plan with the information described above. Project risk areas must be closely monitored by the Project Manager throughout the project.

Examples might include constraints, mandates, scope and complexity of the system, legislation not yet defined or the expectation of team members to perform dual roles.

7 IMPLEMENTATION PLAN AND SCHEDULE

The Implementation Plan Schedule should include a baseline schedule of all tasks that need to be performed for the entire development life cycle of the project. The plan should map all future development of the project, and it should also map to each deliverable of the project.

7.1 WORK BREAKDOWN STRUCTURE

The Work Breakdown Structure (WBS) is an invaluable tool for planning and controlling the project. It is an enumeration of all work activities in hierarchic refinements of detail, which organizes the work to be done into smaller, more manageable component tasks and activities with quantifiable inputs, outputs, and assigned responsibilities. This high granularity in task definition allows time and cost estimates to be prepared based on many small work packages which is more accurate than estimates based on high-level, top-down project requirements.

The initial draft WBS is developed at the end of the Strategy Stage and is refined during the Analysis and Design Stages by the Project Manager with input from the Project Team.

The goal of the WBS is to define all work required to satisfactorily complete the project and to highlight all major milestone events. Task decomposition allows you to assign responsibility for individual tasks and provides specific and verifiable measures of successful completion. Following are additional notes on developing a WBS:

- Development of a WBS is an iterative process proceeding from the general to the specific as uncertainty decreases and details of the project become clear. The first-pass (top-level) WBS is constructed upon completion of the strategy phase.
- The WBS is updated and refined with additional detail during analysis and design and delivered in subsequent Forward Systems Development Plans.
- The high-level system design (identification of major functions, their dependencies, interfaces, etc.) must be completed prior to development of a detailed WBS.
- The WBS establishes a basis for developing detailed estimates, assigning tasks to team members, and monitoring overall progress of the project.
- The WBS must be comprehensive - one of the biggest sources of error with project estimates is omission of overlooked or unexpected activities.

7.2 RESOURCE REQUIREMENTS

From a resource planning perspective, it is critical to understand that the number of resources required by a project varies as a function of project stage and that the activities performed in each stage require specific skills on the part of the resources assigned to them. For example, personnel involved during the front end of the project (strategy, analysis, and design stages) should be senior level personnel (CSA2 or CSA3) experienced in structured analysis and design techniques. Upon completion of the design stage, focus shifts to implementation-related activities in which case coding and development expertise are required.

A resource loading chart is a useful tool for outlining anticipated personnel requirements and should be developed for each project. This information is also useful when evaluating resource requirements of candidate projects against the availability of appropriately skilled personnel to determine what is (and is not) reasonably achievable within existing resource and financial constraints.

A schedule of tasks and activities and milestones should be prepared by the Project Manager with the WBS. Include milestones such as completion of a deliverable or structured walkthroughs.

7.3 CONFIGURATION MANAGEMENT

The intent of this section is to provide developers, project managers, project leaders, and end-user support staff with a framework for implementing Software Configuration Management on software development projects. The document is divided into two sections; the first introduces the discipline of Software Configuration Management (hereon referred to as SCM), describes the major components of SCM, and positions SCM in the context of the application development process, the second section presents guidelines for implementing SCM during the development and production operation of software applications. The application of SCM to the maintenance/enhancement stages of the application life cycle and to the utilization of packaged application software is also discussed.

7.4 SOFTWARE CONFIGURATION MANAGEMENT OVERVIEW

The System Development Methodology (SDM) provides a framework for developing software applications. It outlines the major tasks and deliverables comprising the system development process. Software Configuration Management is an approach to maintaining control over a software project and its resulting deliverable products. All of the items or deliverables produced as part of the development process are collectively referred to as the software configuration.

These development products or configuration items can be divided into three broad categories:

- Software modules or programs. This includes source code, object libraries, and executable files as well as command and batch files.
- Documents that describe the software modules (targeted at both technical staff and end-users)
- Data Structures

7.5 SOFTWARE CONFIGURATION ITEMS

A software configuration item is a formal deliverable produced in accordance with the System Development Methodology. All formal deliverables must be governed by configuration management practices in order to assure both the quality and integrity of the software product and to ensure overall success of the project. All deliverables produced as part of the System Development Methodology should be placed under configuration management control. These include:

- Project Plans
- Strategy Report
- Requirements Specifications
- System Design Reports
- Conversion Specifications
- User's Guides
- Systems Control Documentation
- Transition Plans
- Software Modules
- Data Structure/Database Definitions
- Test Plans
- Test Specifications
- Test Data
- Training Materials

- Development tools and utilities

In summary, Software Configuration Management benefits the application development process by:

- reducing software errors,
- ensuring software integrity,
- ensuring that changes are correctly implemented,
- providing a history for all changes to the software product.

7.6 BASELINING

A baseline is a milestone in the development process that is marked by the delivery, review, and formal approval of a software configuration item (a formal deliverable). Prior to becoming a baseline, informal change control is applied to the deliverable under development. That is, the developer makes changes to the deliverable at his/her own discretion as needed. However, once the item has undergone formal review and is approved, a baseline is created for that deliverable.

7.7 CONFIGURATION STATUS ACCOUNTING

Configuration status accounting provides a complete history of all changes made to the components of the software configuration. It helps provide answers to the following questions:

- What changes were made?
- Who made the changes?
- When were the changes made?
- What else is affected by the changes?

The inputs to configuration status accounting are the configuration items (deliverables) from the Configuration Identification function, change control information from the Change Control process, and status information from the Configuration Audits and Reviews. This information provides complete change Trace-ablity throughout the application life cycle.

7.8 APPLYING SCM TO APPLICATION DEVELOPMENT AND MAINTENANCE

Software Configuration Management and Change Control mechanisms are established by the Project Manager at the beginning of the project and are enforced throughout development of the application and after the application is released for production. The following sections outline practical guidelines for establishing SCM and Change Control practices in the application development and maintenance environments.

The examples provided are specific to the DEC/ORACLE application development platform however, the general concepts on which they are based can be applied to other application platforms.

7.9 THE PROJECT DIRECTORY

This section outlines the recommended organization for all directories and files comprising a software application in the VAX/VMS environment. The proposed directory structure provides a central repository for all project-related files including:

- documentation
- software modules, object libraries, and executables
- command and batch procedures
- production data/test data
- log files
- software tools/utilities
- public access files
- software libraries

The proposed directory structure is intended as a guideline or point of reference for developing project directory structures. Individual directory structures should be governed by the size and complexity of the application under development, the number of developers, and other attributes of the project or application. New subdirectories (and libraries) should be added to the hierarchy as new needs arise. In addition, the project team may be faced with managing multiple versions of a software application or managing both production and development versions of a single release. The project directory should support these types of requirements as well.

The following figure outlines a sample project directory structure for VAX/VMS applications:

Sample VAX/VMS Project Directory Structure

Directory	**Logical Name**
[PROJECT_DIR]	
[.DEV]	
[.DATA]	PROJECT$DEV_DATA
[.COM]	PROJECT$DEV_COM
[.EXE]	PROJECT$DEV_EXE
[.OBJ]	PROJECT$DEV_OBJ
[.WORK]	
[.Developer Username]	PROJECT$WORK (Assigned to developer)
[.Developer Username]	
.....	
[.Developer Username]	
[.PRD]	
[.DATA]	PROJECT$PRD_DATA
[.COM]	PROJECT$PRD_COM
[.EXE]	PROJECT$PRD_EXE
[.PUBLIC]	PROJECT$PUBLIC
[.SCRATCH]	PROJECT$SCRATCH
[.SRCLIB]	PROJECT$SRCLIB
[.CMSLIB]	PROJECT$LOG
	PROJECT$REPORTS

This directory structure can be implemented across multiple disks for purposes of improving performance and/or disk space availability.

7.10 DIRECTORY STRUCTURE DESCRIPTION

The sample directory structure supports two distinct environments:

- The Development Environment
- The Production Environment

This configuration is particularly useful for those projects utilizing an incremental or phased approach in which case the project team must manage both production and non-production software configurations.

The word 'PROJECT' in the following examples represents the name (typically an acronym) of the specific project.

VMS Logical definitions can be used to simplify navigation through the various subdirectories. These Logical names can be defined automatically for each project team member upon login by including the DCL logical name definitions in the LOGIN.COM command file or some other initialization file. In the directory structure outlined in Figure 1, separate logical names are defined for the production and development subdirectories (i.e. PROJECT$DEV_DATA vs. PROJECT$PRD_DATA).

An alternative approach is to define a single logical name (i.e. PROJECT$DATA) and "toggle" it's definition (point to either the production or development subdirectories) depending on the environment in which you wish to work.

- **PROJECT$WORK** - Each team member is provided with a workdirectory in which to do development on project-related deliverables.

- **PROJECT$DEV_OBJ** - A repository for all object libraries(.OLBs) which contain the object modules of frequently called routines.

- **PROJECT$DEV_EXE** - Contains all executable or "run-able" files comprising the system configuration. Files to be maintained here include (but are not limited to) executables (.EXE), object libraries, SQL*Forms files (.FRM), and SQL*PLUS files (.SQL), etc.

- **PROJECT$DEV_COM** - Contains all DCL command procedures that contain the logical name definitions, ACL procedures, etc. for the project's development configuration.

- **PROJECT$DEV_DATA** - Contains all data files (test data, etc.) used during development (excluding ORACLE RDBMS data files).

- **PROJECT$DEV_LOG** - A repository where all batch and error log files can be directed during the development and testing.

- **PROJECT$SRCLIB** - Contains the files comprising the project software Library (DEC/VAX CMS). The project Library provides version control and change trace-ability for all project files.

- **PROJECT$PRD_EXE** - Contains all "runnable" files comprising the project's production configuration. Files to be maintained here include executables (.EXE), SQL*Forms files (.FRM), and SQL*PLUS files (.SQL), etc.

- **PROJECT$PRD_COM** - Contains all DCL command procedures that contain the logical name definitions, ACL procedures, etc. for the project's production configuration.

- **PROJECT$PRD_DATA** - Contains all data files (excluding ORACLE RDBMS data files) required to support the production application.

- **PROJECT$PRD_LOG** - A central repository for all error and log files generated during production operation of the application.

- **PROJECT$PUBLIC** - A general purpose directory for software objects, data files, documents, etc. that are to be made available to the general public.

- **PROJECT$SCRATCH** - A repository for temporary or transient data files, (i.e. sort/merge files, temporary report files, etc.) generated/deleted during the course of processing. This scratch directory alternately be defined as each user's default login (SYS$LOGIN) directory.

- **PROJECT$LOG** - The central repository for all LOG files (.LOG) generated by the application. This directory is a subdirectory within the global LOG directory established for all VAX-resident applications.

- **PROJECT$REPORTS** - The central repository for all report files(.LIS) generated by the application. This directory is a subdirectory within the global REPORTS directory established for all VAX-resident applications.

The files resident in the production area of the directory represent the latest production build or configuration. As such, only the highest versions (most recent) of each file comprising the production configuration should be present in any of the production subdirectories.

While VMS provides an array of security and control facilities and allows multiple versions of the same file to exist simultaneously within a directory, these features do not provide the tracking and reporting mechanisms required of the change control process. Software libraries provide a much more effective means of enforcing version control, restricting access, and providing proper auditing and reporting mechanisms.

7.10.1 CHANGE CONTROL IN THE ORACLE DEVELOPMENT ENVIRONMENT

The Software Configuration Management and Change Control mechanisms described above should also be applied to the software objects produced in the ORACLE development environment.

These include:

CASE*Dictionary/application definitions
SQL*Forms (.INP) files
SQL*Menu (.DMM) files
SQL*ReportWriter (.REP) files
SQL*Plus (.SQL) files

The CASE*Dictionary is the central repository for all information describing the ORACLE application. It maintains complete, up-to-date, and fully cross-referenced information about the complex interdependencies and relationships between the many objects comprising the application.

The CASE*Dictionary Version Control Facility provides a mechanism to support the multiple versions of an application and its database-related information as it evolves over the application life cycle. This facility provides the following capabilities:

- create and delete new versions of the application, freeze (suspend the ability to make changes) and unfreeze specified versions of the application, grant access (to other ORACLE users) to the different versions of an application.

- When an application version is frozen, the ability to insert, update, and delete the individual application objects comprising the applications is disabled for that version.

Only the owner of the CASE*Dictionary application has the ability to create new versions of the application or freeze existing ones and to grant access to the underlying dictionary tables to other ORACLE users. In order to control access to the application resources maintained in the dictionary, it is recommended that these capabilities be restricted to the Configuration Manager for the project. That is, the Configuration Manager should be the owner of the application-related information residing in CASE*Dictionary. With smaller project teams, the Configuration Manager role may be shared by one or more individuals.

Outside of CASE*Dictionary, the ORACLE development environment provides no mechanisms for managing change to individual software objects such as SQL*Forms .INP files, SQL*Plus command procedures, SQL*Menu .DMM files, etc. In order to ensure the quality and integrity of these software modules, it is recommended that each be managed by the CMS project software library.

The project team must coordinate and support a number of different system environments across multiple hardware and software platforms. Each of these environments require their own database and supporting software configuration, including software module files, VMS directories, VMS logical names and data. The specific software environments that need to be supported are described below

1. System Development

The system development environment physically resides on the team's development system. This platform supports day-to-day application development activities comprising system analysis, design, coding and module testing.

2. Testing

System and integration-level testing environment will reside and coexist on the project team's development system with the system development environment. The acceptance testing environment will reside on the DEP VAXcluster system. The testing environments are comprised of software modules and associated test case data required to support the testing process.

3. Training

The training configuration supports end-user training activities. This environment is comprised of the software modules and associated data corresponding to a specific release of application software. The training configuration should closely correspond to the production system configuration in operational use. This system supports end-user training and will contain data that is specific to individual application system lessons and lesson plans. The training configuration resides on the DEP VAXcluser system.

4. Production and Maintenance

Changes are incorporated into the software configuration long after the application is transitioned into production. In fact, experience has shown that the number of change requests increases dramatically in the months immediately following installation of a new application as elusive bugs are uncovered and end-user's identify modifications or new functions to better tailor the application to their day-to-day work-flow. As such, application of formal Change Control throughout the maintenance and enhancement stage is essential to the continued operation and success of the application.

From an administrative standpoint, change requests during the production stage of the application are handled a bit differently than those received during earlier development stages. After the application is transitioned into operation, the Project Manager and project team are typically re-assigned to other projects or development activities. Requests for changes or "bug fixes" are now directed to the centralized development organization where they are reviewed, prioritized, and serviced based on relative priority by the centralized application maintenance staff. The "Request for Services" section of the "User Guide to BIS Services" describes the process for requesting modifications or enhancements to existing centralized DEP applications.

7.11 ACCEPTANCE CRITERION AND TESTING STRATEGY

An initial Test Strategy is developed upon completion of the analysis stage and is refined and updated during the design and build stages by the Project Manager, the Project Team, and the End-User Organization.

Proper testing of the software is critical to the overall quality of the end-product application. Historically, it is also the most neglected or over-looked component of the development process. As such the testing process must be properly planned and methodically executed to ensure acceptable quality standards.

The software testing strategy defines the scope of the testing effort by outlining the types of tests to be performed, when they are to be performed, the resources that are required to support the test effort, and outlines the test-related deliverables that are to be produced.

The initial testing strategy is developed during the analysis stage after system requirements have been refined and stabilized. Test specifications are developed for each software module based on the detailed module design specifications. The test specification outlines test objectives (test cases), the test data to be used, and expected test results.

At a minimum, the Test Plan should address the following:

Module Testing. Focuses on verification of the smallest unit of software design-the module. Using the detailed design specification as a guide, important control paths are tested to uncover errors within the boundary of the module. The man-machine interfaces are tested to assure that information properly flows into and out of the module under test, allowable boundary values are verified, and module-data structure interface is tested to assure that data is properly stored according to established integrity rules. Module testing is performed during the Build Stage.

System Testing. The purpose of system testing (also referred to as integration testing) is to ensure that the system as a whole satisfies input/output specifications and that interfaces between modules/programs/subsystems are correct. Emphasis is placed on system access, security, performance, and recovery capabilities.

Acceptance Testing. Acceptance testing is performed by the client organization with support from the project team to ensure that the application satisfies established acceptance criteria and that both manual procedures and automated functions perform according to stated requirements. Acceptance testing is performed during the Transition Stage.

<Insert>

ATTACHMENTS

(Optional)

<Insert>

Glossary

(Optional)

<Insert>

Index

(Optional)

<Insert>